Remarks/Arguments

These remarks are in response to the Office Action dated August 11, 2005 ("Office Action"). This reply is timely filed. At the time of the Office Action, claims 1-20 were pending in the application. Claims 6, 12, and 18 have been rejected under 35 U.S.C. §112, second paragraph. Claims 1-20 have been rejected under 35 U.S.C. §103(a). The rejections are set out in more detail below.

I. Claim Rejections Under 35 U.S.C. §112

Claims 6, 12 and 18 were rejected under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. In particular, the Examiner notes that in claims 6, 12, and 18, the limitations "the processing step" and "the processing means" lack sufficient antecedent basis. In response to the Examiner's rejection, claims 6, 12, and 18 have now been amended.

II. Brief Review of Applicants' Invention

Prior to addressing the Examiner's rejections on the art, a brief review of applicants' invention is appropriate. The present invention relates to a method and apparatus for high speed interprocess communications ("IPC"). In conventional IPC, multiple processes can communicate with one another via a shared region of random access memory ("RAM") to which each process can write data, and from which each process can read data. When communicating through the shared region of RAM, a first process functioning as a message source can write a message to the shared region of RAM. Subsequently, a second process, functioning as a message receiver, can read the written message from the shared region of RAM. Thus, at a minimum, two system calls are required to move n bytes of data from the first process to the second process through the shared region of RAM. Moreover, 2*n bytes of data will be stored in total: n bytes into the shared region of RAM, and n bytes into user memory space associated with the second process.

To overcome the excessive overhead associated with conventional IPC utilizing a shared region of RAM, the high speed IPC method and apparatus of the present invention avoids moving 2*n bytes of data by passing from the first process directly to a

message list of the second process a memory offset which is used by the second process to access the data. Importantly, the second process then can manipulate and modify the data in place within the shared region of RAM. Accordingly, it is not required that the data be copied from the shared region of RAM to an alternate location while the manipulation takes place, thereby improving system efficiency.

III. Claim Rejections Under 35 U.S.C. §103(a)

Presently, claims 1 through 20 are pending in the subject patent application ("Application"). In the Office Action, however, each of claims 7, 9-12 were rejected as being unpatentable over U.S. Patent No. 6,181,707 to Erickson et al. ("Erickson et al."), in view of U.S. Patent No. 6,754,666 to Brookler, et al. ("Brookler" et al). Claims 1-6, 8, and 13-18 were rejected as being unpatentable over Erickson et al., in view of Brookler, et al., and further in view of U.S. Patent No. 6,148,377 to Carter et al. ("Carter et al."). Finally, claims 19 and 20 were rejected as being unpatentable over Erickson et al., in view of Brookler, et al, in view of Carter et al., and further in view of U.S. Patent No. 5,991,845 to Bohannon et al. ("Bohannon et al.").

Erickson et al. relates to an intercom system that includes a network of peer processor-controlled circuit modules having a plurality of processor circuit modules or cards connected to a common, time-division-multiplexed ("TDM") bus. Each of the processor circuit modules/cards includes a control data communications circuit ("CDCC"), which functions as a communications co-processor. The CDCC includes a means for automating the transmission and reception of simultaneous variable-length control messages among any number of the peer processor-controlled circuit modules using the TDM parallel bus. In moving variable-length data blocks, each card's local microprocessor and each card's CDCC both access data queues from opposite sides of shared dual-port RAM. In sending a message from a source card to a destination card, the source card's microprocessor stores data to a transmit queue. The source card's CDCC transmitter automatically reads the transmit queue and dispatches appropriately-timed data onto the TDM bus towards the destination card.

Referring to independent claims 1, 7 and 13, each of the claims recites that the first process adds to a message list corresponding to the second process a memory offset which corresponds to the location of data in the message buffer. None of the

cited references teach or suggest this limitation. In Erickson et al. the microprocessor of the source card loads an offset into its CDCC's Word-In-Queue Register. This offset corresponds to the offset of the last message word that is stored in dual-port RAM. The CDCC's transmit hardware portion offsets the base address of its transmit queue in dual-port RAM with the Word-In-Queue Register and uses that combined address to fetch the first word of the message from its own transmit queue in dual-port RAM. As the message is sent, the Word-In-Queue is then regenerated at the receiver and the receiver uses this address offset to place the message data into the same-numbered location(s) in its receive queue. (Col. 7, lines 44-55). By loading its Word-In-Queue Register, the source microprocessor triggers the process whereby the CDCC transmit hardware portion automatically dispatches the message data, one word per TDM frame, onto the TDM bus. Such a process requires many more steps and is far more resource intensive than the claimed method of high speed interprocess communication.

Brookler et al. also does not teach or suggest a first process adding a memory offset to a message list of a second process. In fact, Brookler et al. makes no mention of a memory offset, message list, or equivalents thereof. One aspect of the Brookler et al. invention describes a method of efficiently storing data items in a database management system ("DBMS"). The DBMS data is initially stored in the disk drive space (col. 7, lines 50-53). The DBMS includes data stored therein as well as schema and data stored in one or more tables defined in the schema (Id.). Computer-executable process steps of the DBMS are transferred from the disk space via the computer bus to RAM and executed therefrom by the CPU (col. 7, lines 45-49). It is important to note the difference between DBMS process steps and DBMS schema and data. The DBMS schema and data are not being transferred to RAM; only the DBMS process steps are transferred to RAM (Id.). Furthermore, any in-place schema and data manipulation to the DBMS schema and data are occurring exclusively outside of RAM, namely within disk space. Since Brookler et al. does not access the message buffer in shared RAM to manipulate the data contained within the location corresponding to a memory offset, it is for this reason that Brookler et al. does not mention or suggest memory offsets or their use. Accordingly, Brookler et al. is incompatible with the scope of Applicants' claims.

Further still, Brookler et al. discloses two possible approaches for accessing DBMS data stored in the disk space. The first approach is an SQL approach in which SQL language is used to access "disk-based data" that is stored and managed by the DBMS. This continues to indicate that the database data in Brookler et al. is not pointed to a location in RAM using a memory offset as is taught by Applicants' method claims 1 and 7, as well as apparatus claim 13. The second approach is a "memory-based" approach read from the DBMS and then stored in memory. However, Brookler et al. admits that such a memory-based approach has the disadvantage that is requires "substantially more memory than the first approach." (Col. 16, lines 15-25). The second approach would indicate at the very least 2*n bytes of data would be stored, a problem that goes to the crux of Applicants' invention.

Carter et al. also fails to make up for the deficiencies found in Erickson et al. and Brookler et al regarding Applicants' independent claims 1 and 13. Carter et al. discloses distributed shared memory systems and processes that can connect into each node of a computer network. The system encapsulates the memory management operations of the connected nodes and provides an abstraction of a shared virtual memory that can span across each node of the network. Optionally, the shared memory can span across each memory device connected to a computer network. Accordingly, each node on the network having the distributed shared memory system of the invention can access the shared memory.

However, Carter et al. also does not teach or suggest a first process adding a memory offset to a message list of a second process as is recited in Applicants' claims 1, 7, and 13. Carter et al. fails to satisfy the express "memory offset" language of Applicants' claims. Instead, the data in Carter et al. is passed by value and not by reference from process to process. For example, Carter et al. teaches that the system can include a coherent replication controller for generating a copy, or select number of copies, of a portion of the addressable memory space maintained in the local persistent memory device of a first computer and for storing the copy in the local persistent memory device of a second computer. (Col. 3, lines 41-46). The Applicants explicitly state, "Processes are notified of the location of the message data rather than actually receiving a copy of the message data." (Applicants' specification, page 8, lines 18-19).

Therefore, the Carter et al. reference fails to teach the passage of data by reference between processes with the use of memory offsets.

In contrast, the second process of Applicants' claimed invention can access the shared data in RAM with the use of memory offsets. Independent method claims 1 and 7 both recite that the first process adds to the message list of the second process a memory offset corresponding to the location in the message buffer. Further still, independent apparatus claim 13 recites a "means for said first process to add to said message list of said second process a memory offset corresponding to said location in said message buffer." Consequently, claims 1, 7 and 13 are distinguishable from the cited art. Claims 2-6, 8-12 and 14-20 are believed to be allowable at least by virtue of their dependence on allowable base claims. In the event that the Examiner cannot produce a specific reference in Erickson et al, Brookler et al., and Carter et al., to the addition of the "memory offset" to a "message list" of a "second process," the independent and dependent claims of the Application are *prima facie* patentable.

IV. Conclusion

Applicants have made every effort to present claims which distinguish over the prior art, and it is believed that all claims are in condition for allowance. Nevertheless, Applicants invite the Examiner to call the undersigned if it is believed that a telephonic interview would expedite the prosecution of the application to an allowance. In view of the foregoing remarks, Applicants respectfully request reconsideration and prompt allowance of the pending claims.

Respectfully submitted,

11-1-05
Date

Robert J. Sacco
Registration No. 35,667
SACCO & ASSOCIATES, P.A.
P.O. Box 30999
Palm Beach Gardens, FL 33420-0999
Tel: 561-626-2222